بسم الله الرحمن الرحيم
آلة إليريا نموذجية Modular Ilyria Machine

1. **لغة آلة إليريا**:
   1. لغة آلة ذات تعليمات محددة
   2. تتألف التعليمة من رقم طبيعي
   3. طول هذا الرقم يكون حسب مساحة الذاكرة المتوفرة
   4. الرقمان المتواجدان على اليسار في التعليمة يمثلان العملية
   5. باقي الأرقام تمثل عنوان الذاكرة الذي تتم عليه العملية
   6. لا يهم أن تمثل التعليمة برقم عَشري أو رقم سادس عشري
   7. تنقسم التعليمة إلى فئتين:
      1. تعليمة تنفيذية: التي تقوم بالعمليات على المتغيرات أو تغير عداد البرنامج أو تعلم ذاكرة على أنّها ملصق
      2. تعليمة حجز: التي تحجز مكان في الذاكرة لمتغير ما
   8. مثال:
      1. **100004+**
         1. 10+ تمثل العملية
         2. **0004** تمثل الذاكرة التي تتم عليها العملية

الشيفرة	الاسم	name
01	ثماني	byte
02	قصير	short
03	حرف	char
04	صحيح	int
05	طويل	long
06	عائم	float
07	مزدوج	double
08	حروف	chars

10	اقرا	read
11	اكتب	write
12	اكتبسج	writenl

20	حمل	Load
21	خزن	store

30	جمع	add
31	طرح	sub
32	قسمة	div
33	ضرب	mul
34	باقي	mod
35	زد	inc
36	نقص	dec

40	تقاطع	branch
41	تقاطعسلب	branchneg
42	تقاطعصفر	branchzero
43	انهاء	halt

50	اس	exp
51	نوع	caltype
52	دقة	prc
53	جذر	sqrt

60	ملصق	label

2. **وحدة جلب وفك تشفير التعليمة:**
    1. تجلب التعليمة الحالية من الذاكرة:
        1. مكان التعليمة يحدده عداد البرنامج
        2. تفكك التعليمة إلى عملية ومعامل
        3. تضع العملية في سجل العملية
        4. تضع المعامل في سجل المعامل
        5. تغير قيمة عداد البرنامج لمكان التعليمة التالية
3. **وحدة التنفيذ:**
    1. تقوم بتنفيذ التعليمة:
        1. تحدد العملية حسب سجل العملية
        2. تنفذ العملية على الذاكرة المحددة في سجل المعامل
    2. هناك عمليات لا تحتاج إلى معامل:
        1. إيقاف
4. **حجز المتغيرات:**
    1. يقوم بحجز الذاكرة للمتغيرات
    2. هناك 8 أنواع للمتغيرات:

1. **ثماني byte** يحجز 1 بايت من الذاكرة
2. **قصير short** يحجز 2 بايت من الذاكرة
3. **حرفي char** يحجز 2 بايت من الذاكرة
4. **صحيح int** يحجز 4 بايت من الذاكرة
5. **طويل long** يحجز 8 بايت من الذاكرة
6. **عائم float** يحجز 4 بايت من الذاكرة
7. **مزدوج double** يحجز 8 بايت من الذاكرة
8. **نصي** سلسلة من **حرفي**
9. يحجز كل متغير عدد البايتات اللازمة حسب نوعه
10. النوع **النصي** ثابت ولا يتغير
11. طريقة الحجز
    1. قسم التعليمة إلى أجزاء الفاصل بينها هو الفراغ
    2. قم بتحويل الجزء الأول إلى رقم صحيح
    3. في حالة كان الرقم الصحيح الموافق للجزء الأول:
        1. **01+** : احجز ثماني في الذاكرة الخاصة بالمتغيرات
        2. **02+** : احجز قصير ...
        3. **03+** : احجز حرفي ...
        4. **04+** : احجز صحيح ...
        5. **05+** : احجز طويل ...
        6. **06+** : احجز عائم ...
        7. **07+** : احجز مزدوج ...
        8. **08+** : احجز نصي ...

5. **مجمع إليريا:**
    1. يحول التعليمات المكتوبة بلغة تجميع إليريا إلى الشيفرات والذواكر الموافقة لها
        1. مثال:

            ثماني 0 **ث**

            01+ 0 **4**

6. **السجلات:**
    1. المجمع: فيه تتراكم النتائج الصادرة عن العمليات كالجمع والطرح والقسمة… وأيضا عند تحميل قيمة متغير من ذاكرة معينة فإن هذه القيمة تنسخ وتوضع في المجمع
    2. شيفرة العملية: تضع فيها وحدة جلب وفك الشيفرة قيمة تمثل العملية الحالية التي ستجرى على المعامل
    3. المعامل: فيه يخزن مكان الذاكرة التي ستجرى عليها العملية الحالية
    4. سجل التعليمة: تخزن فيه التعليمة الحالية
    5. عداد البرنامج: يحدد مكان التعليمة التالية في الذاكرة
    6. نهاية التعليمات: عنوان ذاكرة يمثل حد الذاكرة المحجوزة للتعليمات طبعا وكذلك بداية أول ذاكرة مخصصة للمتغيرات
        1. أول عنوان ذاكرة خاص بالتعليمات هو 0

7. بداية المتغيرات: أول عنوان ذاكرة مخصص للمتغيرات وحد الذاكرة المحجوزة للتعليمات
8. بداية الإجراءات: أول عنوان ذاكرة مخصص للإجراءات وكذلك حد الذاكرة المخصصة للمتغيرات
1. حد ذاكرة الإجراءات هو حد الذاكرة الرئيسية

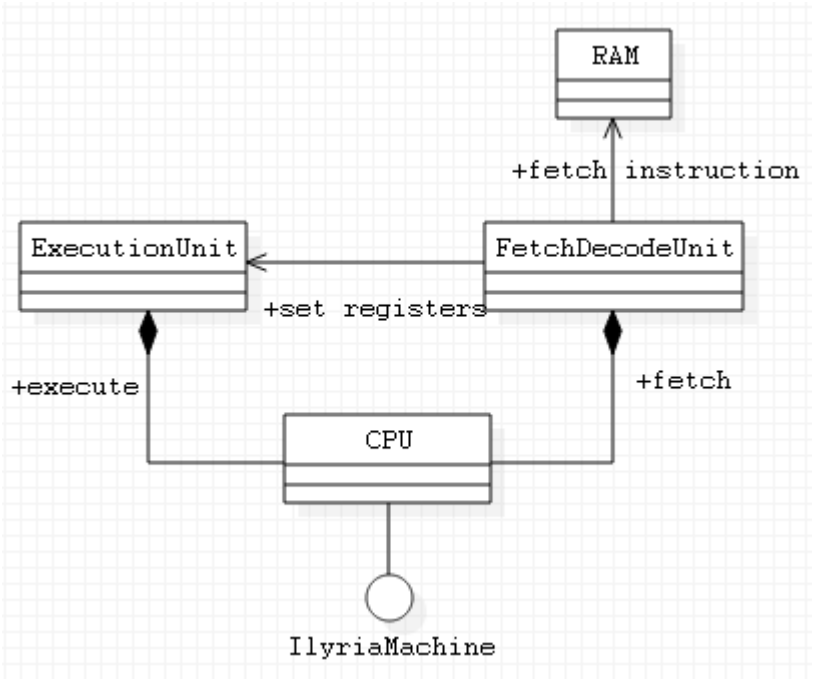
7. الأسماء المرشحة لتكون فئات:

الاسم	جافا
لغة آلة إليريا	IlyriaMachine
وحدة المعالجة المركزية	CPU
وحدة جلب وفك الشيفرة	FetchDecodeUnit
وحدة التنفيد	ExecutionUnit

وحدة الذاكرة	Memory

مجمع إليريا	IlyriaAssembler
لغة تجميع إليريا	IlyriaAssembly

8. نموذج الفئات:



***IlyriaMachine*«*interface*»**

BYTE : Integer = 1 {ReadOnly}
SHORT : Integer = 2 {ReadOnly}
CHAR : Integer = 3 {ReadOnly}
INT : Integer = 4 {ReadOnly}
LONG : Integer = 5 {ReadOnly}
FLOAT : Integer = 6 {ReadOnly}
DOUBLE : Integer = 7 {ReadOnly}
CHARS : Integer = 8 {ReadOnly}

READ : Integer = 10 {ReadOnly}
WRITE : Integer = 11 {ReadOnly}
WRITENL : Integer = 12 {ReadOnly}

LOAD : Integer = 20 {ReadOnly}
STORE : Integer = 21 {ReadOnly}

ADD : Integer = 30 {ReadOnly}
SUB : Integer = 31 {ReadOnly}
DIV : Integer = 32 {ReadOnly}
MUL : Integer = 33 {ReadOnly}
MOD : Integer = 34 {ReadOnly}
INC : Integer = 35 {ReadOnly}
DEC : Integer = 36 {ReadOnly}

BRANCH : Integer = 40 {ReadOnly}
BRANCHNEG : Integer = 41 {ReadOnly}
BRANCHZERO : Integer = 42 {ReadOnly}
HALT : Integer = 43 {ReadOnly}

EXP : Integer = 50 {ReadOnly}
CALTYPE : Integer = 51 {ReadOnly}
PRCISION : Integer = 52 {ReadOnly}
SQRT : Integer = 53 {ReadOnly}

LABEL : Integer = 70 {ReadOnly}
FUNCTION : Integer = 71 {ReadOnly}
RETURN : Integer = 72 {ReadOnly}
CALL : Integer = 73 {ReadOnly}

+fetch() : Integer
+decode() : Integer[]

+execute() : Integer *+reset()*

CPU implements *IlyriaMachine*
-nextId : Integer -cpuId : Integer {ReadOnly} #«create»fetchDecode : FetchDecodeUnit #«create»execution : ExecutionUnit #«create»ram : ByteBuffer #«create»input : Scanner {ReadOnly} #«create»output : PrintStream {ReadOnly}
+CPU«constructor»() +fetch() : Integer +decode() : Integer[] +execute() : Integer +reset() +getInput() : Scanner +getOutput() : PrintStream

FetchDecodeUnit
#execute : ExecutionUnit #ram : ByteBuffer
+FetchDecodeUnit«constructor»(memories : ByteBuffer, execute : ExecutionUnit) +fetch() +decode()

ExecutionUnit
#accum : Double #operationCode : Integer #operand : Integer #calculationType : Byte

```
#precision       : Byte

#instruction    : integer
#programCounter : Integer

#dataIndex : integer;

-INSTRUCTION_LENGTH : Byte {ReadOnly}

-END_INSTRUCTION : Integer {ReadOnly}
-END_DATA        : Integer {ReadOnly}
-END_MEMORY      : Integer {ReadOnly}

#ram : ByteBuffer;

#input  : Scanner     {ReadOnly}
#output : PrintStream {ReadOnly}
```

```
+ExecutionUnit«constructor»( memory : ByteBuffer)
+execute()
```

```
+getAccumulator() : Double
+setAccumulator(value : Double)

+getOperationCode() : Integer
+getOperand()       : Integer
+setOperationCode(operation : Integer)
+setOperand(operand : Integer)

+getCalculationType() : Byte
+getPrecision()       : Byte
+setCalculationType(type : Byte)
+setPrecision(precision : Byte)

+getInstructionRegister() : Integer
+getProgramCounter()      : Integer
+setInstructionRegister( instruction : Integer )
+setProgramCounter( pc : Integer )
+incrementProgramCounter()

+getEndOfInstructions() : Integer
+getEndOfData() : Integer
+getEndOfMemory() : Integer
```

+reset()

IlyriaAssembler
cpu : CPU input : Scanner path : Path output : PrintStream values : HashTable< String integer >
+IlyriaAssembler«constructor»(source : Path) -readNextLine() : String -splitLine(line : String) -compile() : Integer

9. بالنسبة لخطوات التنفيذ هي مفصلة في النص المصدري للمشروع ivm2
10. إضافات(لم تجسد بعد):
    1. تعليمة الحجز الديناميكي ALLOC:
        1. التي تحجز مساحة جديدة للمتغيرات
        2. ثم تضيف لـ dataIndex عدد البايتات المحجوزة
    2. تعليمة الحذف DEL:
        1. التي تحرر مساحة من الذاكرة
        2. ثم تنقص من dataIndex عدد البايتات المحذوفة
        3. (لا داعي لها)مشكلة إزاحة جميع عناوين المتغيرات
        4. (تم حلها في الإجراءات)مشكلة تغيير جميع التعليمات التي ترجع إلى هذه العناوين
11. الإجراءات:
    1. تحدد الإجراءات بشيفرة العملية PROCEDURE
    2. ينتهي الإجراء بالتعليمة RETURN:
        1. هذه التعليمة ترجع إلى العنوان التالي بعد تعليمة الإستدعاء
        2. لا ترجع أي قيمة بل التعليمات التي في الإجراء هي التي تقوم بهذا…
    3. تستدعى الإجراءات بشيفرة العملية CALL
        1. أولا يقفز عداد المكدسة إلى العنوان التالي في المكدسة
        2. يضع علامة بداية الإطار بالتعليمة FRAME
        3. يضع التعليمات الواحدة تلوى الأخرى
        4. بعد وضع التعليمة RETURN التي يجب أن تنتهي بها الإجراءات:
            1. يعطى للتعليمة RETURN عنوان التعليمة التالية بعد الإستدعاء
            2. يقفز عداد البرنامج إلى التعليمة الأولى للإجراء الحالي
            3. ينفذ التعليمات وقد يحجز أماكن في الذاكرة
            4. محو المتغيرات المحلية يدوي
            5. يجب أن تحذف المتغيرات المحلية بشكل عكس المصرح به

6. يجب أن يحذف الإطار
5. في حالة استدعاء إجراء لإجراء آخر يحدث نفس الشيء